Module 2 - PHP to Laravel

Module 1 Topics

In Module 1, we learned:

1. **PHP** as a server-side application language
2. **MySQL** for database storage
3. **REST API** through Request & Response
4. **Bearer Token** for secure REST APIs

Module 1 Issues

Challenges from Module 1:

1. **Low-level PHP** → requires too much detail for proper development
2. **Complex SQL** → hard to write and manage
3. **REST API coupling** → difficult to implement and maintain
4. **Bearer Token setup** → requires extra, manual work

Module 2 as the Solution

Laravel solves these problems:

1. **High-level framework** → abstracts complexity and simplifies development
2. **Eloquent ORM** → replaces SQL with intuitive PHP functions
3. **REST API made simple** → MVC scaffolding builds APIs with minimal code
4. **Built-in security** → middleware makes secure APIs straightforward

Automation with Scripts

In Module 2, we also **automate builds**:

- **Windows Users**:
  - Use **WSL2 + Ubuntu (HW1)**
  - Convert scripts with `dos2unix` before running
- **Linux/Mac Users**:
  - Use the same scripts without modification
- Result: Everyone gets **identical builds** across systems

Module 2 Goals

By the end of Module 2, students will:

1. Use the **Laravel framework** instead of raw PHP
2. Use **Eloquent ORM** instead of raw SQL
3. **Port Project 1 REST APIs** into Laravel-based REST APIs

Module 2 Structure

Same structure as Module 1:

1. **Lecture** → `lecture/` directory
2. **PDF slides** → `pdf/` directory
3. **Code (scripts & source)** → `code/` directory
4. **Questions** → `questions/` directory

Preparation for Module 2

Windows Users

1. Install WSL2 & Ubuntu (HW1)
2. Run `dos2unix` before executing any script
3. Use `php artisan serve --host=0.0.0.0` to start Laravel
4. After setup, same workflow as Linux users

Installation

Visit: `module2/code/1_Laravel/2. Laravel Installation and Project Structure`

1. Run **`mac_install.sh`** (Mac) or **`wsl2_install.sh`** (Windows/Linux)
   - Installs all tools automatically
   - See professor if installation fails
2. Verify by running **`run1-6.sh`** in:
   `module2/code/1_Laravel/6. Making REST APIs/`
   - Copy `run1-6.sh` + `student-api1-6/` to a temp directory (`~/temp/ase230`)
   - Run `dos2unix run1-6.sh` (WSL2 only)
   - Run: `bash run1-6.sh`
   - Confirm all APIs build & run automatically

Running Laravel Server

1. Move into the generated project directory:

```
cd student-api
```

2. Start the server:

```
php artisan serve
# or for WSL2:
php artisan serve --host=0.0.0.0
```

Testing API Endpoints

Basic tests:

```
curl http://localhost:8000/api/test
curl http://localhost:8000/api/hello
curl http://localhost:8000/api/hello/YourName
```

Bearer Token tests:

```
curl -H "Authorization: Bearer $(cat api_token.txt)" \
http://localhost:8000/api/goodbye
```

You are ready to start the Module 2 & Project 2